REMARKS

Claims 1-26 were pending in the application. Claims 1, 3, 4, 10, 11, 13, 14, 16, 17, 23, 24, and 26 were rejected under 35 U.S.C. § 102(a) as anticipated by U.S. Patent 6,108,715 ("Leach"). Claims 2 and 15 were rejected under 35 U.S.C. § 103(a) over Leach in view of U.S. Patent 5,590,327 ("Biliris"). Claims 5 and 18 were rejected under 35 U.S.C. § 103(a) over Leach in view of U.S. Patent 5,504,901 ("Peterson"). Claims 6, 7, 19, and 20 were rejected under 35 U.S.C. § 103(a) over Leach in view Peterson and further in view of U.S. Patent 6,381,735 ("Hunt"). Claims 8 and 21 were rejected under 35 U.S.C. § 103(a) over Leach in view of U.S. Patent 6,263,379 ("Atkinson") and further in view of U.S. Patent 5,592,588 ("Reekes"). Claims 9 and 22 were rejected under 35 U.S.C. § 103(a) over Leach in view of U.S. Patent 6,336,148 ("Doong"). Claims 12 and 25 were rejected under 35 U.S.C. § 103(a) over Leach in view of U.S. Patent 5,838,971 ("Stadler").

The applicant has amended claims 1, 13, 14, and 26. New claims 27-28 have been added. Support for the amendments and the new claims can be found, for example, on page 10 lines 10-31. No new matter has been added. In view of the foregoing amendments and the following arguments, the applicant respectfully submits that all claims are in condition for allowance.

1.   **Response to Rejections under Section 102**

Claims 1, 3, 4, 10, 11, 13, 14, 16, 17, 23, 24, and 26 were rejected as anticipated by Leach. The applicant respectfully traverses the rejection.

As amended, claim 1 recites a method in which a first process is run in a first address space, and a second process is run in a second address space. The first process includes a request to send data to the second process. The data has a data type. A type creation function is called at runtime to create a type object that describes the data type. The type object has a set of associated functions for processing data. The associated functions include a marshalling function that encodes data and an unmarshalling function that decodes data. The marshalling function takes as inputs the type object and the data. The data is communicated to the second process, and the unmarshalling function in the second process decodes the encoded data.

As explained in the specification (page 1, lines 11-19), a process's address space is the region of memory to which the process has access. Because the first process and the second process are run in two different address spaces, the processes cannot communicate simply by reading and writing into common memory.

A type object for the data type is dynamically generated by calling a type creation function at runtime. The data type of the data can be, for example, integer, real, string, pointer, structure, or array (page 2, lines 17-18). The type object describes the format of the data type, and the marshalling function uses the dynamically generated type object to encode the data (page 2, lines 7-14).

Leach discloses a method and system for invoking remote procedure calls. When a client process needs to send data to a remote procedure, an operating system creates a stack for the remote procedure and maps the stack into the operating system's address space. The operating system then copies parameters directly from the client's stack to the remote procedure's stack. After the parameters are copied, the remote procedure executes using the parameters on the remote procedure's stack. The operating system can also copy parameters from the remote procedure's stack to the client's stack. Abstract. Leach's method makes marshalling unnecessary in most situations. Column 3, lines 27-30.

Leach does not disclose or suggest calling a type creation function at runtime to create a type object that is input to a marshalling function. Instead, Leach's method maps stacks in memory and copies data from stack to stack to avoid marshalling data. Formatting of parameters during the transfer is done according to a table in a kernel of the operating system, which contains signatures for each remote procedure. Leach's parameters are not formatted according to type objects that are created dynamically by calling type creation functions. For at least the foregoing reasons, claim 1 is allowable over Leach.

Claim 10 recites the method of claim 1 in which the type creation function is also called in the second process to create a second instance of the type object. Because Leach does not disclose or suggest calling a type creation function or creating a type object, claim 10 is allowable over Leach.

Claims 3, 4, and 11 depend directly from claim 1, and are allowable for at least that reason.

Independent claims 13, 14, and 26 include limitations corresponding to those of claim 1. Claims 16, 17, 23, and 24 depend directly from claim 14, and are allowable for at least that reason.

2.   Response to Rejections under Section 103

Claims 2 and 15 were rejected as unpatentable over Leach in view of Biliris. The applicant respectfully traverses the rejection. Claim 2 depends directly from claim 1, and claim 15 depends directly from claim 14. The examiner does not appear to contend that Biliris suggests or discloses calling a type creation function at runtime to create a type object that is input to a marshalling function. Claims 2 and 15 are therefore allowable for at least the same reasons as claim 1.

Claims 5 and 18 were rejected as unpatentable over Leach in view of Peterson. The applicant respectfully traverses the rejection. Claim 5 depends indirectly from claim 1, and claim 18 depends indirectly from claim 14. The examiner does not appear to contend that Peterson suggests or discloses calling a type creation function at runtime to create a type object that is input to a marshalling function. Claims 5 and 18 are therefore allowable for at least the same reasons as claim 1.

Claims 6, 7, 19, and 20 were rejected as unpatentable over Leach in view of Peterson and further in view of Hunt. The applicant respectfully traverses the rejection. Claims 6 and 7 depend indirectly from claim 1, and claims 19 and 20 depend indirectly from claim 14. The examiner does not appear to contend that Peterson or Hunt suggests or discloses calling a type creation function at runtime to create a type object that is input to a marshalling function. Claims 6, 7, 19, and 20 are therefore allowable for at least the same reasons as claim 1.

Claims 8 and 21 were rejected as unpatentable over Leach in view of Atkinson and further in view of Reekes. The applicant respectfully traverses the rejection. Claims 8 and 21 require that encoded data is generated by a marshalling function, and that the encoded data

includes an encoded representation of a type object description, which describes the format of a data type.

Atkinson discloses storing a flag in the marshalled data that indicates why the data was marshalled. Atkinson does not disclose or suggest including an encoded representation of a type object description in encoded data.

Reekes discloses storing flags in a description for an object, where the flags indicate, for example, what kind of data the object can accept as an input, or what kind of data the object outputs. Reekes does not disclose or suggest including in encoded data an encoded representation of a type object description, which describes the format of a data type. For at least the foregoing reasons, claims 8 and 21 are allowable over the combination of Leach, Atkinson, and Reekes.

Claims 9 and 22 were rejected as unpatentable over Leach in view of Doong. The applicant respectfully traverses the rejection. Claim 9 depends directly from claim 1, and claim 22 depends directly from claim 14. The examiner does not appear to contend that Doong suggests or discloses calling a type creation function at runtime to create a type object that is input to a marshalling function. Claims 9 and 22 are therefore allowable for at least the same reasons as claim 1.

Claims 12 and 25 were rejected as unpatentable over Leach in view of Stadler. The applicant respectfully traverses the rejection. Claim 12 depends indirectly from claim 1, and claim 25 depends indirectly from claim 14. The examiner does not appear to contend that Stadler suggests or discloses calling a type creation function at runtime to create a type object that is input to a marshalling function. Claims 12 and 25 are therefore allowable for at least the same reasons as claim 1.

3.   **Interview Summary**

The undersigned representative thanks Examiner Moslehi for the courtesy of a telephone interview on March 2, 2004. The subject of the interview was how the Examiner is reading the limitations of claim 1. The Examiner explained that he is reading claim 1 in terms of two DLLs passing data to each other. He said that the two DLLs would operate in different address spaces

Applicant : Brandyn Webb   Attorney's Docket No.: 07844-465001 / P429
Serial No. : 09/781,007
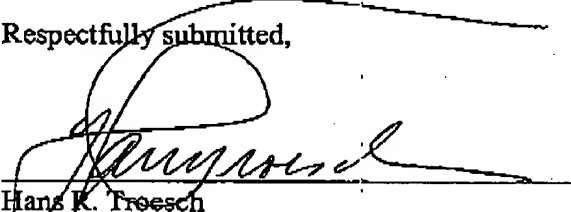Filed   : February 9, 2001
Page   : 15 of 15

because they occupy different parts of memory. He also said that the first type object and the marshalling function limitations could be read very generally and requested more specific information about them. The Examiner pointed out that transferring data between socket objects requires defining data types. The applicant explained that in the claim a type creation function is called to create an object that describes a data type and that is used to encode data having the data type. The applicant also explained that in the claim the type creation function is called at runtime. The Examiner observed that object-oriented languages, e.g., Objective-C and Smalltalk, commonly create objects at runtime.

4.   **Conclusion**

The applicant submits that all claims are in condition for allowance and asks that all claims be allowed. Please apply any charges or credits to deposit account 06-1050.

Respectfully submitted,

Date: _3 Mar 04_

Hans K. Troesch
Reg. No. 36,950

Fish & Richardson P.C.
500 Arguello Street, Suite 500
Redwood City, California 94063
Telephone: (650) 839-5070
Facsimile: (650) 839-5071

50196654.doc